A COMPREHENSIVE ANALYSIS OF ARTIFICIAL SPEECH SYNTHESIS TECHNOLOGIES, THEIR CLASSIFICATION, AND EXISTING APPROACHES AND ALGORITHMS FOR DISTINGUISHING HUMAN SPEECH FROM SYNTHETIC SPEECH

S.U.Nasirov
sultan250593@gamil.com
Tashkent University of Information Technologies named after Muhammad al-Khorezmi

Abstract: This article traces the history of speech synthesis from Wolfgang von Kempelen's 18th-century mechanical speech device - often overshadowed by his deceptive "Mechanical Turk" - to modern zero-shot text-to-speech (TTS) systems. Unlike early mechanical synthesizers that required manual manipulation, contemporary TTS uses large-scale, multi-speaker datasets, neural audio codecs, and language models to generate speech in unseen voices from short reference samples. The article also highlights key applications of TTS, including assistive communication (e.g., Stephen Hawking), hands-free reading, and human-computer interaction.

Keywords: Speech synthesis, Text-to-Speech, Zero-shot TTS, Wolfgang von Kempelen, Neural audio codec, Assistive communication

The problem of speech synthesis has a significantly longer history than the field of speech-to-text conversion. In 1769, in Vienna, Wolfgang von Kempelen created the famous "Mechanical Turk" automaton for Empress Maria Theresa, which was designed to play chess. The device consisted of a wooden cabinet behind which a mannequin in the form of a robot was positioned, capable of moving chess pieces through a mechanical arm. The "Turk" was exhibited throughout Europe and America for several decades, reportedly defeating Napoleon Bonaparte and even playing a match against Charles Babbage. Had it not been secretly operated by a human chess player concealed inside the cabinet, the machine could have been regarded as one of the earliest achievements in the history of artificial intelligence.

Von Kempelen's lesser-known yet scientifically significant invention was the first full-sentence speech synthesizer, developed between 1769 and 1790. Unlike the "Mechanical Turk," this device was not a deceptive exhibition but rather a genuine experimental attempt to mechanically replicate the process of human speech production. The system incorporated an air chamber functioning as lungs, a rubber oral cavity and nasal passage, a reed mechanism simulating the vocal cords, whistles for generating fricative sounds, and an auxiliary air chamber providing pressure for the production of plosive sounds. By manipulating a set of specialized levers with both hands, the operator could open and close various apertures and alter the configuration of the flexible leather "vocal tract," thereby producing a range of consonant and vowel sounds.

More than two centuries later, speech synthesizers are no longer based on wood, leather, and manually operated mechanical components. Modern speech synthesis systems have eliminated the need for human operators and rely on advanced computational methods to generate speech automatically. Text-to-Speech (TTS), also referred to as speech synthesis, is the process of converting written text into spoken language and can be regarded as the inverse of Automatic Speech Recognition (ASR). While ASR transforms speech signals into textual representations, TTS generates a corresponding speech waveform from a given text input.

TTS technology has found widespread application across a variety of practical domains. It plays a crucial role in spoken language interaction systems, text-reading applications, computer games, and

assistive communication technologies designed for individuals who have lost their ability to speak. A notable example is the astrophysicist Stephen Hawking, who relied on speech synthesis technologies for communication after losing his natural voice due to amyotrophic lateral sclerosis (ALS). In this section, TTS algorithms trained on large-scale speech datasets are described in a manner analogous to the ASR algorithms discussed in the previous section. Furthermore, brief attention is given to other important application areas of speech technologies and their broader impact on human–computer interaction.

The primary objective of the text-to-speech (TTS) task is to generate a speech waveform corresponding to a given text while taking into account specific voice characteristics selected or specified by the user. Historically, TTS systems were typically developed by collecting hundreds of hours of speech recordings from a single speaker in controlled laboratory environments and training a large model exclusively on that data. As a result, the synthesized system was capable of generating speech only in that particular speaker's voice. Producing speech in a different voice required the collection of a new speech dataset from another speaker and the retraining of the entire system. Modern approaches differ substantially from this paradigm. Contemporary TTS systems are trained as speaker-independent synthesizers using tens of thousands of hours of speech data collected from thousands of speakers. Consequently, they are capable of generating speech in voices that were not encountered during training. To synthesize speech in a new target voice, only a short speech sample from the desired speaker is required. In such systems, the input typically consists of a text prompt together with a brief speech recording - often approximately three seconds long - representing the voice characteristics that the synthesized speech should emulate.

This approach is known as zero-shot text-to-speech (TTS), as the system is capable of synthesizing speech in a voice that may not have been encountered during the training phase. Modern TTS systems achieve this capability by employing language modeling and conditional generation techniques. In this process, large-scale speech datasets are first transformed into discrete audio token sequences using an audio tokenizer based on a neural audio codec. Subsequently, a language model is trained with a vocabulary that incorporates both text tokens and speech tokens.

During the training stage, the model receives two types of input sequences: a text transcript and a short speech sample from the target speaker. Both textual and speech data are converted into discrete token representations. The model is then trained to conditionally generate speech tokens corresponding to the input text while preserving the vocal characteristics of the target speaker. As a result, the system learns to synthesize speech that accurately conveys the semantic content of the text while reproducing it in an acoustic form that matches the specified voice.

During the inference stage, the language model receives the tokenized text together with a reference speech sample representing the desired voice. The speech sample is converted into discrete audio tokens using an audio codec. Subsequently, the model generates the corresponding speech tokens through a conditional generation process. In the final stage, these generated tokens are transformed back into an acoustic waveform. As a result, an artificial speech signal is produced that conveys the content of the input text while reflecting the characteristics of the selected voice.

Text-to-Speech (TTS) applications are software systems that utilize speech synthesis technologies to convert written text into spoken language. Such applications provide numerous benefits to users across various domains. For example, they enable individuals to listen to articles, books, and other written materials while driving, exercising, or engaging in activities that require continuous visual attention. Consequently, TTS technologies improve accessibility, enhance user convenience, and facilitate hands-free access to information.

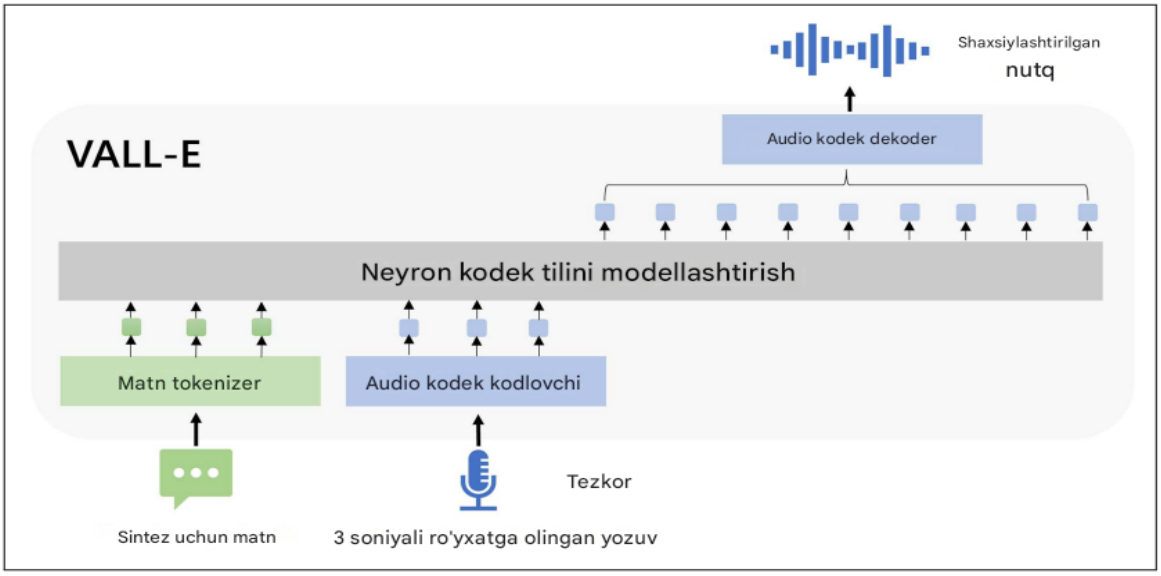

Figure 1.2. Architecture and Operational Framework of the VALL-E Neural Speech Model

TTS applications are also valuable tools for individuals learning a foreign language or seeking to improve their pronunciation skills. Such systems can be adapted to read texts aloud in multiple languages, thereby assisting learners in developing their listening comprehension and pronunciation abilities. In addition, users can customize various speech parameters, including speaking rate, intonation, and pronunciation characteristics, according to their specific needs and preferences.

Text-to-Speech technologies are primarily employed to facilitate natural human–computer interaction, enable voice-based control of software applications, and enhance the usability of user interfaces. However, many operating systems are proprietary products, limiting users’ and developers’ access to their source code. As a result, integrating speech processing and recognition technologies into open-source projects can be challenging, particularly when compatibility with proprietary platforms and services is required.

Furthermore, there is currently no comprehensive and centralized information resource that systematically summarizes the advantages and limitations of open-source speech recognition systems. As a result, selecting the most appropriate speech recognition solution for a specific application remains a challenging task. The primary objective of this research is to reduce the costs associated with selecting suitable systems for commercial and scientific applications, facilitate the effective use of open-source speech recognition technologies, and provide recommendations regarding both speech recognition and text-to-speech implementation. Within the scope of this study, several widely used open-source speech recognition systems, including CMU Sphinx, HTK, iAtros, Julius, Kaldi, and RWTH ASR, were selected and evaluated. The comparison was conducted according to several criteria, including recognition accuracy, computational performance, ease of use, and internal architectural design.

One of the key advantages of the CMU Sphinx system is the availability of pre-trained language and acoustic models for several languages. In addition to standard English, the system supports languages such as Russian, Kazakh, and several others. CMU Sphinx is distributed under the BSD license, which permits its integration into commercial applications. Although its recognition accuracy is generally somewhat lower than that of Kaldi, it remains suitable for commercial deployment and provides an application programming interface (API) that facilitates the development of third-party applications. The HTK system primarily supports the English language and is distributed under the HTK license, which allows access to and distribution of the source code. The toolkit is mainly

intended for research and experimental purposes and has been widely used in the development and evaluation of speech recognition algorithms.

Among the systems evaluated in this study, iAtros demonstrated comparatively lower usability. The system supports English and Spanish languages; however, its major limitation is that it operates exclusively on Linux-based operating systems. Furthermore, iAtros is distributed under the GPLv3 license, which restricts its incorporation into proprietary commercial software without disclosing the source code. Consequently, the system is generally considered less suitable for commercial applications.

The Julius speech recognition system is implemented in the C programming language and provides both a command-line interface and an application programming interface (API) for integration with third-party software. The system supports English and Japanese languages and is distributed under a BSD-like license. Julius can also be recommended for educational purposes, as it combines several advantages of the HTK toolkit while additionally offering support for Japanese speech recognition.

The Kaldi speech recognition toolkit incorporates a wide range of algorithms designed to reduce the dimensionality of acoustic feature representations and improve overall system performance. Developed in C++, Kaldi benefits from the efficiency and computational speed associated with this programming language. The system is built upon a modular architecture, which facilitates customization, the addition of new functionalities, and the correction of existing issues. However, Kaldi provides only a command-line interface, making its integration into third-party applications more challenging compared to systems that offer dedicated APIs. Kaldi primarily supports the English language and is distributed free of charge under the Apache License. This licensing model allows the toolkit to be incorporated into commercial products without the requirement to disclose the source code, making it an attractive solution for both research and industrial applications.

Similar to iAtros, the RWTH ASR system can employ Gaussian Mixture Models (GMMs) during the acoustic modeling stage. A distinctive feature of this system is its ability to utilize phonetic and acoustic properties of speech signals during feature extraction. RWTH ASR is implemented in C++ and is based on a modular architecture, which facilitates system customization and further development. The toolkit is distributed under a specialized license that permits its use exclusively for non-commercial purposes. Consequently, RWTH ASR is not suitable for integration into commercial applications. Nevertheless, the system can be effectively applied in tasks where recognition accuracy is of primary importance and computational time is not a critical constraint. The performance of speech recognition systems is commonly evaluated using the Word Error Rate (WER) and Word Recognition Rate (WRR) metrics. WER measures the proportion of incorrectly recognized words, whereas WRR represents the percentage of words that are correctly recognized by the system. These evaluation metrics are calculated using the following formulas:

$$WER = \frac{S + I + D}{T} \quad (1.1)$$

$$WRR = 1 - WER \quad (1.2)$$

In these formulas, S denotes the number of word substitution errors, I represents the number of word insertion errors, D indicates the number of word deletion errors required to restore the original utterance from the recognized sequence, and T corresponds to the total number of words in the reference utterance. These performance metrics are typically expressed as percentages. To evaluate recognition speed, the Real-Time Factor (RTF) metric is commonly employed. This metric represents the ratio between the time required to process and recognize a speech signal and the actual duration

of the speech signal itself. It is also referred to as the Speed Factor. The metric is calculated using the following formula:

$$\text{WER} = \frac{T_{sav}}{T} \quad (1.3)$$

In this formula, T_{sav} denotes the time required for speech recognition, while T represents the duration of the input speech signal. This metric is expressed as a fraction of real time and is used to evaluate the computational efficiency of a speech recognition system. Based on the analysis of Text-to-Speech (TTS) systems presented in this section, it can be concluded that speech synthesis technologies provide significant convenience by converting written information into spoken form. Such applications are capable of reading texts in multiple languages while allowing users to customize speech parameters, including speaking rate, intonation, and pronunciation, according to their individual preferences and requirements. In particular, TTS technologies serve as effective assistive tools for individuals with visual impairments, users who experience reading difficulties, and language learners seeking to improve their listening comprehension and pronunciation skills.

Table 1.2.

Comparative Analysis of Text-to-Speech (TTS) Systems

System	WER%	WRR	SF
CMU Sphinx (pocketsphinx/sphinx4)	21,4 / 22,7	78,6 / 77,3	0,5 / 1
HTK	19,8	80,2	1,4
iAtros	16,1	83,9	2,1
Julius	23,1	76,9	1,3
Kaldi	6,5	93,5	0,6
RWTH ASR	15,5	84,5	3,8

Automatic speech synthesis is a technology designed to convert textual input into spoken language. The quality of the synthesized speech is of critical importance, as it largely determines the feasibility of employing speech synthesis technologies in modern commercial and practical applications. At present, automatic speech synthesis systems are regarded as software tools capable of transforming written texts and other forms of information into audible speech. In the English-language literature, such systems are commonly referred to as Text-to-Speech (TTS) systems, which generate speech output from textual input.

References

[1] Delgado H., Evans N., Kinnunen T., Lee K. A., Liu X., Nautsch A., Patino J., Sahidullah M., Todisco M., Wang X., Yamagishi J. ASVspoof 2021: Automatic Speaker Verification Spoofing and Countermeasures Challenge. 2021.

[2] McFee B., Raffel C., Liang D., Ellis D. P. W., McVicar M., Battenberg E., Nieto O. librosa: Audio and Music Signal Analysis in Python // Proceedings of the 14th Python in Science Conference. – 2015. – pp. 18–25.

[3] Povey D., Ghoshal A., Boulianne G., Burget L., Glembek O., Goel N., Hannemann M., Motlicek P., Qian Y., Schwarz P., Silovsky J., Stemmer G., Vesely K. The Kaldi Speech Recognition Toolkit // IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). – 2011.

[4] Bishop C. M. Pattern Recognition and Machine Learning. – New York: Springer, 2006.

[5] Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge: MIT Press, 2016.

[6] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. – 2nd ed. – O'Reilly Media, 2019.

[7] Murphy K. P. Machine Learning: A Probabilistic Perspective. – Cambridge: MIT Press, 2012.